PROCEEDINGS

55. IWK

Internationales Wissenschaftliches Kolloquium
International Scientific Colloquium

13 - 17 September 2010

Crossing Borders within the ABC

Automation,

Biomedical Engineering and

Computer Science

Faculty of
Computer Science and Automation

www.tu-ilmenau.de

TECHNISCHE UNIVERSITÄT
ILMENAU

Impressum
Published by

Publisher:	Rector of the Ilmenau University of Technology Univ.-Prof. Dr. rer. nat. habil. Dr. h. c. Prof. h. c. Peter Scharff
Editor:	Marketing Department (Phone: +49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de) Faculty of Computer Science and Automation (Phone: +49 3677 69-2860) Univ.-Prof. Dr.-Ing. habil. Jens Haueisen
Editorial Deadline:	20. August 2010
Implementation:	Ilmenau University of Technology Felix Böckelmann Philipp Schmidt

USB-Flash-Version.

Publishing House:	Verlag ISLE, Betriebsstätte des ISLE e.V. Werner-von-Siemens-Str. 16 98693 Ilmenau
Production:	CDA Datenträger Albrechts GmbH, 98529 Suhl/Albrechts
Order trough:	Marketing Department (+49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de)

ISBN: 978-3-938843-53-6 (USB-Flash Version)

Online-Version:

Publisher:	Universitätsbibliothek Ilmenau ilmedia Postfach 10 05 65 98684 Ilmenau

Home / Index:
http://www.db-thueringen.de/servlets/DocumentServlet?id=16739

HIGHER GENERALIZATION PERFORMANCE OF ARTIFICIAL NEURAL NETWORKS WITHOUT REDUCING LARGE DATA BASES

M. Schneider, F. Calcagno, M. Stieglitz, U. Lehmann, J. Krone

Institut für Computer Science, Vision und Computational Intelligence,
Fachhochschule Südwestfalen, Frauenstuhlweg 31, 58644 Iserlohn
Tel. +49 (0)2371 566303 Fax +49 (0)2371 566209
E-Mail: {MSchneider, Lehmann, Krone}@fh-swf.de
{Fabrizio.Calcagno, Marcel.Stieglitz}@stud.fh-swf.de

ABSTRACT

This proceeding presents a possibility to enhance the accuracy of artificial neural networks. The database is divided in smaller parts and will be sorted. For each part an artificial neural network (ANN) will be trained. The overall result is calculated by the arithmetic mean value of all partial results from the trained ANNs. Also large databases, which usually can't be trained because of lack of resources, may be trained with this method. Consequently, the training time was accelerated and the total error was minimized, without reducing the data base and more ANNs.

1. INTRODUCTION

At the training of artificial neural networks (ANNs) by using a large data base of complex processes is often the problem, that the trained ANN has a good generalization only for partial data. Furthermore, the time required for the training of a large number of data sets and according on training algorithm increases huge. A lesser amount of data per ANN reduces the training time and increases the model precision. This can be achieved by reducing the data base, where data reduction is as well always associated with data loss. This can be prevented if the existing data base is divided into several sub models, which together form a complete model.

2. MODELLING CONCEPT

2.1. Data Handling

Before training, the database has to be prepared appropriate for the training. For this purpose is based on the case, that the input database exists of N rows and M columns.

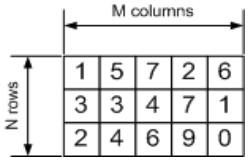

Figure 1 Database exists of N rows and M columns

N = Number of data sets

M = Number of input neurons

For each of the M columns the input database is copied. Each of this copy will be sorted ascending after one of the M columns. Because the sorting is arranged only by one column per copy, the order of the data in the other columns is also be changed. To obtain this order, for each sorting a whole copy of the database is being produced. [4]

With this method, M sorted input databases are generated. How such a sorted database could look like, is shown in figure 2. Each of the sorted input databases will be separated into D parts. The border areas are observed separately. Because, each of the input databases is sorted ascending against one of the M columns, all extreme values dedicated to this column are in the border areas. It is recommended to determine the input-database and choosing the size of the border areas to ensure that all extreme values of the input-database of the particular column M get into this border areas.

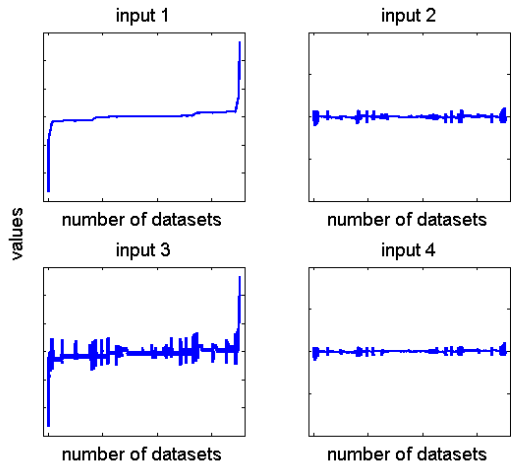

Figure 2: Sorting of a database with separation

The remaining data will be separated into the parts from D=1 and D=n-2. The figure 3 shows how such a separation would look like.

The size of each partition (D) have to be chosen onto the architecture of the ANN and the used hardware. The size of D affects also the duration of training of each ANN, because with a smaller D there are more data sets per part and with a bigger D there are less data sets per part.

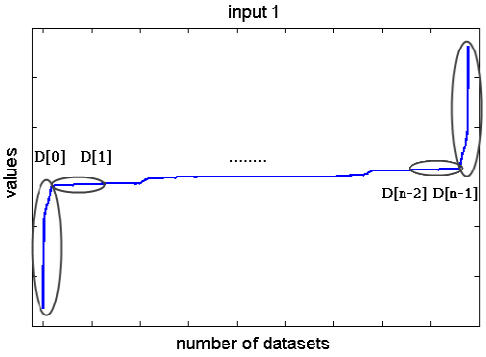

Figure 3: Separation of the database

Additionally, at separation and sorting of the input database for each column of the output database a own ANN may be trained. For each output neuron several expert networks are trained which are specialized to a particular part of the database. The number of ANNs to be trained, can be calculated as follows:

$$Sum\ of\ ann = D \times I \times O \qquad (1)$$

D = Number of data subsets

I = Number of input neurons

O = Number of output neurons

In this way there will be ensured that each ANN is specialized at a specific column of the database.

Example:

The input database exists of a matrix with the dimensions 200,000x20 and the output database exists of a matrix with the dimensions 200,000x10. A single ANN would have 20 input and 10 output neurons. If the database is separated like it has been explained in this chapter, 20 databases are being produced. In this example will be act that D = 10. This means that each database will be separated in 10 parts. Additionally, the output matrix has 10 columns. Each of this columns corresponds one output neuron und could be observed individually. With reference to (1): 10 * 20 * 10 = 2000 ANNs were trained.

In figure 4 is shown an outline of the method.

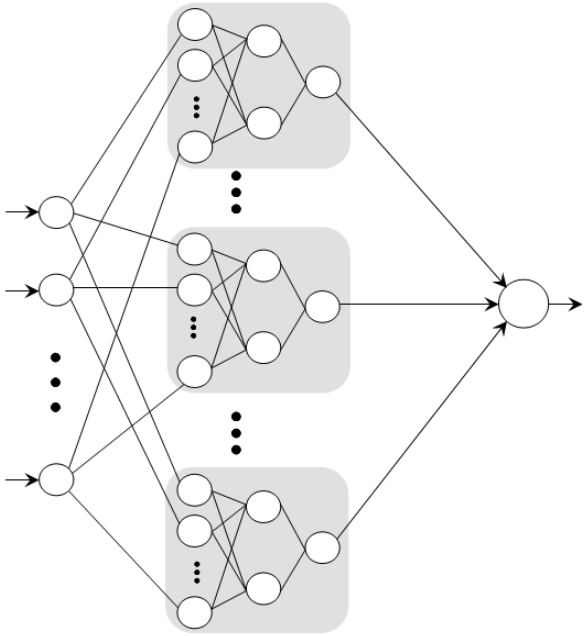

Figure 4: Outline of the method

2.2. Training

The Levenberg-Marquardt-Algorithm [5] that is implemented in the software "Matlab" was used as training method [3]. The learning efficiency of this training method is compared to the Backpropagation-Algorithm [1] significantly higher.

In figure 5 is shown the performance of the Levenberg-Marquardt-Algorithm and the Backpropagation-Algorithm at a training of 10 minutes.

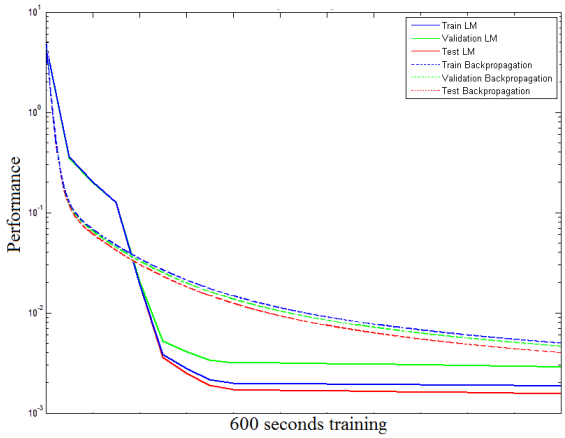

Figure 5: Comparison of the performance plot

The Levenberg-Marquardt-Algorithm has iterated 22 epochs in 10 minutes and the Backpropagation-Algorithm has iterated 2.006 epochs. The Levenberg-Marquardt-Algorithm has been at 120 seconds the same performance as the Backpropagation-Algorithm at 600 seconds.

The performance plot of the Levenberg-Marquardt-Algorithm in figure 5 have a better gradient as opposed to the performance plot of the Backpropagation-Algorithm.

Architecture

In the chapter "Data-Handling" can be seen, that the data sharing leads to more data subsets. For the appropriate selection of neurons for each hidden layer, the number of data sets of each data subsets must be ascertained.

The number of data sets of each data subset may be different, consequently the number of neurons on each hidden layer varies.

Activation function

The activation function on the hidden layer is Tanges Hyperbolicus. The Tangens Hyperbolicus is a differentiable (nonlinear) function. [2]

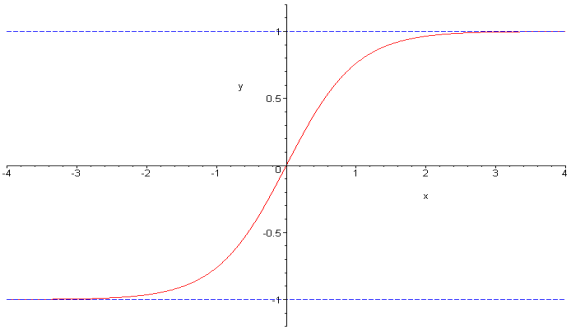

Figure 6: Activation function Tangens Hyperbolicus

The linear activation function is used for the input and output layer.

Memory management

The memory management is a very important aspect for the simulation. With the software "Matlab", the trained data can be stored in so-called "mat-files". The "mat-file" is the default format to save variables in Matlab. For a quick and efficient access to the networks in the simulation, a "mat-file" should be created for each output neuron. In each "mat-file" the networks are stored into individual structures. With this method the loading of all networks is not necessary and therefore memory-friendly in the simulation.

2.3. Simulation

The data set which is to be simulated, must first be classified in a data subset. For each sorting must be established, in which part the data set pertained. In this way an ANN is selected per sorting. Per sorting a simulation takes place for each output neuron. Therefore, the number of results of each output neuron corresponds to the number of input neurons respectively of sorting.

Results

On more than one sorting, for each output neuron is calculated more than one result. In this section the method for grouping of the individual results is described. For each output a result vector is produced. This result vector is sorted ascending. After sorting, particular the first two and the last two results be removed. When simulating was evident that the calculated result is closer to reality. The final result is calculated using the arithmetic mean of the remaining results. The removal of extreme values can be dynamically tuned for the particular database.

Speed optimization

Depending on how many input neurons and output neurons are used, a large number of trained networks cannot be avoided. In a simulation more than one data set could been tested. For each data set the network for each sort is loaded accordingly for each output neuron. So the main memory reaches quickly its limit and the simulation time increases, because the operating system begins to swap data into virtual memory. Basically there are three methods to carry out the simulation:

1. All ANNs loaded into main memory

The maximum speed is reached when all ANNs are completely loaded into main memory. Initially a lot of time for loading the ANNs is required by the slow hard disk access. Following many data sets can be simulated very quickly. The biggest disadvantage from this method is the huge memory requirements.
It is possible that the main memory is running at full capacity even with very small ANNs, because the number of ANNs can be large. At 100 % used capacity of main memory, the ANNs swapped partially out to hard disk.

The simulation times will be drastically increased, because slow disk access take place. This method is recommended only when all networks fit completely into the main memory.

2. ANNs unloaded immediately after use

When the ANNs are unloaded immediately after use, only very little memory is required. However, with this method for each data set the ANNs have to be reloaded. This creates a large overhead, because at each data set as many ANNs have to be loaded from hard disk, as output neuron are present. Therefore this method is the method with the lowest memory requirements, but also the method which claimed the most time because of the large amount hard disk accesses.

3. Combination of method 1 and 2

The most suitable is a method in which the memory is used up and the hard disk accesses are reduced as much as possible. Before an ANN is loaded from the hard disk into memory, first the workspace of Matlab examines whether the ANN is already loaded. If the ANN cannot be found in the Matlab workspace, it is loaded. If it is already loaded, the ANN is used immediately and avoid a hard disk access. In addition, for each ANN the time of loading is stored into the workspace.

Is the main memory working to capacity, the longest in the main memory loaded ANN will be removed, before a new ANN is loaded. In this way, space is made for the next ANN and an outsourcing of the ANNs to the hard disk is prevented. The memory is now fully utilized and the hard disk accesses are reduced.

3. RESULTS AND COMPARISON

This chapter are presented a comparison between the training method which was explained in this proceeding and a method in which was used one ANN for the total data base. For both methods was used the Levenberg-Marquardt-Algorithm.

In figure 7 is shown the mean absolute error about all output neuron for the total data base, it has been sorted in ascending order to better present the MAE.

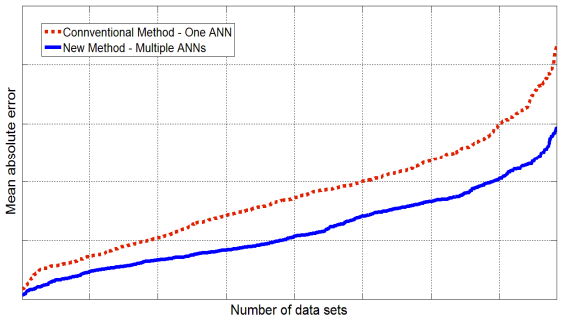

Figure 7: Comparison between ANN and multiple ANNs

The figure 7 shows that many small ANNs provide a better result than a single large ANN. The following formula shows the calculation of the mean absolute error:

$$MAE = \frac{1}{N}\sum_{n=1}^{N}|E_n| \qquad (2)$$

N = Number of outputs per pattern
E = Error of each output per pattern
MAE = Mean absolute error

In figure 8 can be seen the mean absolute error from each output neuron. The figure shows, that an improvement is noted for the most outputs.

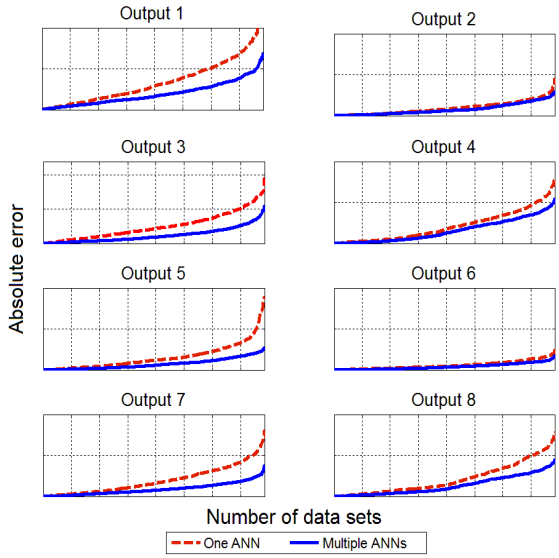

Figure 8: Comparison between ANN and multiple ANNs - All outputs

Of course it may happen that there is only a very minimal difference between the two methods, such as output 2 or output 6. The reason is, that the error already on the two outputs, which was trained with the method of one large ANN, lies in the tolerance range and thus already has a good generalization. However, is seen an improvement at such parts. The following table shows the improvements for each output in percent. The improvement was calculated using the average error over all errors of each method.

Output 1	39,53 %	**Output 2**	25,56 %
Output 3	56,15 %	**Output 4**	34,14 %
Output 5	52,97 %	**Output 6**	38,37 %
Output 7	52,89 %	**Output 8**	37,67 %

Table 1: Improvement of all output neurons

The following formula was used to calculate the improvement:

$$AI = 1 - \frac{MAE(E_{mANN})}{MAE(E_{ANN})} \qquad (3)$$

E_{mANN} = Error of all patterns per output neuron from the method of multiple ANNs

E_{ANN} = Error of all pattern per output neuron from the method of one ANN

MAE = Mean absolute error

AI = Average improvement

With the table can be seen even more the improvement to each output neuron. Additionally, the training time could be shortened by 42 %.

4. CONCLUSION

The comparison shown in chapter 3 makes clear, that the ANNs trained with the method explained in this proceeding, have a better performance than other ANNs, trained with the whole database without sorting and separating.

With this method, it is possible to train large databases which are normally difficult to train, because of physical barriers, such as the size of the main memory, are set.

5. OUTLOOK

Dynamic separation of database

The so far introduced method, separated the database only static among constant given borders. Using a dynamic separation this may be advanced. For example, the ascending slope between all data sets may be calculated. All data sets located into a specific interval of ascending slopes may be put into an own data subset. With a large interval only less databases arise while using a small interval many databases arise.

Overlapping databases

For Example, if there is a huge break between the values the data sets of an database and the database is separated in such a way that this break is not considered, the ANNs have to be extrapolate at this break. It would be imaginable to overlap the generated data subsets. This means that each data subsets get data sets from the adjacent data subsets excepting the data subsets at the boundary. In this way, the whole data area will be covered.

6. ACKNOWLEDGEMENT

This work is being arranged at the Institute for Computer Science, Vision and Computational Intelligence of the South Westphalia University of Applied Sciences in coverage of the research project "Neuroadaptiver Bauplatz im Flugzeugbau" financed by the Federal Ministry of Education and Research.

7. REFERENCES

[1] R. Rojas: *Theorie der neuronalen Netze*, Springer-Verlag Berlin/Heidelberg, 1993

[2] A. Zell: *Simulation neuronaler Netze*, Oldenbourg, Wissenschaftsverlag, 2003

[3] W. Schweizer: *Matlab kompakt,* Oldenbourg, Auflage 4, 2009

[4] *Neuronale Netze. Optimierung durch Lernen und Evolution,* Springer Berlin, 1997

[5] P. K. Shukla, *Levenberg-Marquardt Algorithms for Nonlinear Equations, Multi-objective Optimization, and Complementarity Problems (Operations Research),* Shaker Verlag, 2010